Semantics Enhanced Services: METEOR-S, SAWSDL and SA-REST

Amit P. Sheth, Karthik Gomadam, Ajith Ranabahu
Services Research Lab, kno.e.sis center, Wright State University, Dayton, OH
{amit,karthik, ajith}@knoesis.org

Abstract

Services Research Lab at the Knoesis center and the LSDIS lab at University of Georgia have played a significant role in advancing the state of research in the areas of workflow management, semantic Web services and service oriented computing. Starting with the METEOR workflow management system in the 90's, researchers have addressed key issues in the area of semantic Web services and more recently, in the domain of RESTful services and Web 2.0. In this article, we present a brief discussion on the various contributions of METEOR-S including SAWSDL, publication and discovery of semantic Web services, data mediation, dynamic configuration and adaptation of Web processes. We finally discuss our current and future research in the area of RESTful services.

1 Overview

Our body of research can be divided into three major phases. The first phase related to the METEOR (for "Managing End To End OpeRations") system [6] focused on workflow management and addressed issues of formal modeling, centralized as well as distributed scheduling and execution (including exception handling, security, survivability, scalability and adaptation). The work yielded two notable frameworks:1)WebWork [7], a Web based implementation and 2) ORBWork, a CORBA based implementation. The METEOR project initally started at BellCore in 1990 and was continued at the LSDIS lab until 1998. A commercial spinoff, Infocosm, inc. and the product METEOR EAppS (for Enterprise Application Suite) are other notable accomplishments.

Adopting to the SOA and semantic Web evolution, METEOR evolved into METEOR-S where S stands for services (or Service oriented Architecture) and semantics. It was largely carried out at LSDIS Lab during later 1990s and 2006. One of the significant contributions of METEOR-S research is the submission of WSDL-S specification as a W3C member submission, along with IBM. In 2006, the W3C created a charter for the Semantic Annotation of Web Services (SAWSDL; www.w3.org/ 2002/ws/sawsdl), which used WSDL-S as its primary input. SAWSDL became a W3C candidate recommendation in January 2007.

Our third phase recognizes emergence of Web2.0 and the People Web along with use of microformats for associating metadata to Web resources, and so called light weight web services (RESTful services and WebAPIs). This phase started in 2006 and significantly expanded at the Services Research Lab in Kno.e.sis Center (where our group of 11 researchers moved from the LSDIS lab) in 2007. One of the key initial outcome is a microformat for annotating service descriptions in HTML called hREST and a faceted extension called SA-REST.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Both hREST and SA-REST are in their early stages of research. Further information about these is available at `http://knoesis.wright.edu/research/srl/projects/hRESTs/`.

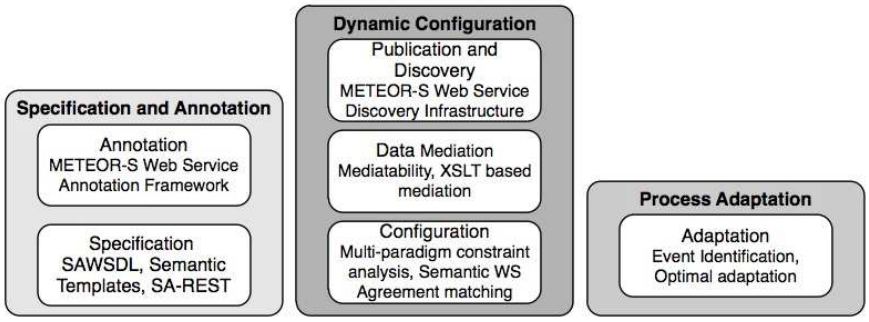

Figure 1: Overview of the various components of the METEOR-S framework.

While other prominent semantic Web service efforts in OWL-S and WSMO have focused on creating service ontologies and process composition to a large extent, the objective of METEOR-S is to define and support the complete life-cycle of Semantic Web processes. METEOR-S adopts an evolutionary approach towards semantic Web services, by extending current SOA (WS-*) standards and specifications to support semantics. We identify three main stages in the life-cycle as illustrated in Figure 1. The first stage comprises of techniques to define annotation mechanisms to extend current SOA standards. Using the annotations to support enhanced discovery and publication of services along with the support for configuration and data-mediation is addressed in the second stage. The third stage addresses identifying events and adapting to various events during execution. We are currently working on exciting area of RESTful services and Web 2.0. More specifically, our research focusses on specifying RESTful services, finding them and integrating them to create smart mashups or smashups. In this article, we present a detailed description of the contributions of the METEOR-S project and briefly describe our current research direction in the area of service oriented computing.

2 Specification and Annotation

The building blocks of SOA-based solutions are self-describing Web services that can be reused across various applications. The Web Service Description Language (WSDL) was created specifically for this purpose and describes the data elements, operations and message bindings. However, WSDL descriptions are not sufficient for the client to unambiguously decipher each operations intended purpose as well as the intended content of its parameters. SAWSDL (which evolved from WSDL-S, first proposed in [13]) overcomes the above limitation by adding semantic meta-data to WSDL elements [15]. Semantic annotations are added to WSDL elements using the *modelreference* extensibility attribute. The *modelreference* value of a WSDL element contains a reference to a concept in the ontology that defines the semantics of that element. This allows service providers to better describe their interfaces and allows clients to better understand the interface descriptions. The original ideas for WSDL-S and SAWSDL were founded on the four types of semantics for services - 1) data semantics: descriptions of the data elements of a service, 2) functional semantics: descriptions of the various operations and functional capabilities of a service, 3) non-functional semantics: descriptions of the non-functional requirements and guarantees, and 4) execution semantics: descriptions of events and faults and how to handle them [10]. We discuss the impact of semantic annotations in realizing dynamic configuration in the next section.

3 Dynamic Configuration

Our research has demonstrated the value of semantic annotations in realizing dynamic SOA environments. The METEOR-S middleware discussed in [5] demonstrates a SOA middleware that supports run time discovery and binding of partner services. Service requirements are both functional and non-functional. Service discovery selects partner services that fulfill the functional requirements. From this set, partners that fulfill the non-functional requirements are selected using constraint analysis.

3.1 Discovery and Publication

Selection of partner services that fulfill the functional requirements of a client is the first step to realize dynamic SOA environments. The METEOR-S Web Service Discovery Infrastructure (MSWDI) is a peer to peer framework for efficiently discovering partner services [12]. MWSDI uses an ontology-based approach to organize registries, enabling semantic classification of all Web services based on domains. Each of these registries supports semantic annotation of the Web services, which is used during discovery process. MWSDI defines four kinds of peers

1. An operator peer controls a Web Service Registry. The role of the Operator peer is to control a registry and to provide Operator services for its registry. The Operator peer also acts as a provider for the Registries Ontology to all other peers who need it.

2. A gateway peer acts as an entry point for registries to join MWSDI. It is responsible for updating the Registries Ontology when new registries join the network. It is also responsible for propagating any updates in the Registries Ontology to all the other peers. Gateway peer is not associated with any registry.

3. Auxiliary peers act as providers of the Registries Ontology.

4. The Client peers are transient members of the peer-to-peer network, as they are instantiated only to allow users to use the capabilities of the MWSDI.

3.2 Multi-Paradigm Constraint Analysis

Partners that fulfill the functional requirements may not fulfill the non-functional requirements. Selecting partners who also fulfill the non-functional requirements is the second step. Non-functional requirements are typically modeled as Service Level Agreements (SLAs). The SLAs lack semantic metadata and are often very generic, thus making it hard to match SLAs from two services. In [9], the authors present a framework to enhance WS-Agreement with structure and semantic metadata. The framework includes a well defined XML based syntax for expressing and semantically annotating SLAs. The additional semantic information allows one to incorporate rules and enables the system to make better matches dynamically.

Non-functional requirements themselves can either be quantitative (*supply time* $\leq$*5 days*) or non-quantitative (*Security must be RSA*). To deal with both, we proposed a multi-paradigm constraint analysis in [1]. The constraint analyzer uses integer linear programming based techniques for optimizing the quantitative constraints and SWRL and SPARQL based techniques for non-quantitative constraints.

3.3 Data Mediation

One of the key benefits of SAWSDL is the systematic approach to data mediation using XSLT. Rather than using XSLT's to mediate between message instances and schemas, SAWSDL advocates mediation at the level of ontologies. To translate a service schema to an ontology, SAWSDL specifies two key techniques - 1) lifting schema mapping and 2) lowering schema mapping. Lifting schema mapping is an XSLT transformation to

convert a service schema to an ontology schema. Lowering schema mapping converts an ontology schema into a service schema. To achieve mediation between two service schemas, the source schema is lifted to its corresponding ontology schema. Using schema transformation techniques, the lifted schema is translated into the target ontology schema. This is lowered into the target service schema. The systematic approach offers a huge upgrade in defining and reusing transformation functions [8].

4 Adaptation

The adaptation phase addresses the problem of adapting business processes to runtime events and faults. These include system events such as service unavailability, as well as business level events such as shipment delays. Creating a middleware system with the ability to monitor and adapt to both types of events can be viewed as a two-step problem. The first step is to identify and subscribe to the events to which the system might need to adapt. The second step is to adapt to those events as and when they occur. We present an approach to automatically identify events that may impact the execution of a process in [3] building upon our research in the area of semantic associations for discovering events from a functional and a non-functional ontology.

Once the events are identified, we address adaption as a stochastic decision making problem using Markov Decision Processes(MDP) [14]. MDP policies are generated by using the events identified, event probabilities described by partner services and the cost impact of the events as described in the SLAs. Further, a cost penalty for adaptation is calculated by considering the constraints that occur across services (inter-service dependencies). This allows us to ensure that the adaptation does not violate the process optimality requirements. We discuss three approaches- 1) a centralized approach in which a central controller maintains all the MDP state information, 2) a decentralized approach in which each MDP acts independently of the other in deciding the optimal action and 3) a hybrid approach in which the decentralized MDPs communicate with each other via the central controller.

5 Beyond SOAP: RESTful Services and Web 2.0

Lately, the RESTful services paradigm has gained a lot of traction. Web applications (such as maps and payment processing) and data (such as news feeds) are being exposed as services that can be invoked using scripting languages such as Ruby, PHP and Javascript. Web application hybrids or mashups have emerged as a very popular way for integrating RESTful services. Despite their popularity, the programming complexity and the fundamental problem of data mediation make it hard for non-expert developers to create meaningful mashups. Our research in the area of RESTful services addresses this limitation. We break down our approach into three steps: 1) specification, 2) finding the right set of services and 3) Service integration.

In the area of specification, we are advocating a new microformat called hRESTs for service descriptions in HTML. hRESTs provides constructs to markup operations and data elements in an API description. hRESTs evolved from our current work on SA-REST, first proposed in [11] and inherited the operation and data element constructs of SA-REST. Furthermore, in addition to operations and data elements, RESTful API descriptions have other facets such as data formats (JSON, GData), and client library bindings (Java, PHP). These are captured using the constructs of SA-REST, which is being modeled as an extension to hRESTs. A more detailed description of hRESTs can be found at: http://knoesis.wright.edu/research/srl/projects/hRESTs/.

RESTful services are often described as Web APIs using HTML. The lack of a model like WSDL makes it difficult to use conventional service discovery approaches. Currently general purpose search engines such as Google are often used for finding these APIs. API search frameworks such as programmableWeb rely on user classification and often yield poor results. In our research, we use traditional text classification techniques for faceted classification and indexing of APIs. We also have developed a ranking algorithm similar to PageRank called Service Utilization (ServiUt rank) for ranking APIs [2]. Finally, in the area of integration, we currently

focus on the problem of data mediation. Though there have been numerous attempts to realize automatic mediation, there is still considerable amount of human effort required in the process. We define a metric called Mediatability that estimates the amount of human effort needed in mediation. The mediatability computation algorithm is a two pass algorithm that uses the concept of nearest common parent, first proposed by Tarjan. The first pass is a top down pass that computes the matching values and the similarity of the two schema trees. The second pass is a bottom up pass that computes the mediatability values using the matching and similarity values [4].

6 Conclusions

In addition to proposing newer techniques and standards, the METEOR-S research has also contributed open source software for handling SAWSDL object models (SAWSDL4J, Woden4SAWSDL), semantic annotation (Radiant) and for discovery and publication (Lumina). Much of the past work in the area of semantic Web services has focused on the WS-* implementation of SOA. Lately, the RESTful approach to SOA has gained popularity, largely due to its lightweight approach. We are currently working on the specification, search and integration of RESTful services and Web APIs. It is our belief that our current research would ease the task of creating mashups and would allows users to create customizable and dynamically configurable smart mashups.

Acknowledgements: We acknowledge the contributions of Professor John Miller, Dr. Kunal Verma and other members of the METEOR-S project at LSDIS lab.

References

[1] R. Aggarwal, K. Verma, J. A. Miller, and W. Milnor. Constraint driven web service composition in meteor-s. In *IEEE SCC*, pages 23–30, 2004.

[2] K. Gomadam, A. Ranabahu, M. Nagarajan, K. Verma, and A. P. Sheth. A faceted classification based approach to search and rank web apis. In *ICWS*, page To Appear., 2008.

[3] K. Gomadam, A. Ranabahu, L. Ramaswamy, A. P. Sheth, and K. Verma. A semantic framework for identifying events in a service oriented architecture. In *ICWS*, pages 545–552, 2007.

[4] K. Gomadam, A. Ranabahu, L. Ramaswamy, K. Verma, and A. P. Sheth. Mediatability: Estimating the degree of human involvement in xml schema mediation. In *ICSC*, page To Appear., 2008.

[5] K. Gomadam, K. Verma, A. P. Sheth, and J. A. Miller. Demonstrating dynamic configuration and execution of web processes. In *ICSOC*, pages 502–507, 2005.

[6] N. Krishnakumar and A. P. Sheth. Managing hetergeneous multi-system tasks to support enterprise-wide operations. *Distributed and Parallel Databases*, 3(2):155–186, 1995.

[7] J. A. Miller, D. Palaniswami, A. Sheth, K. Kochut, and H. Singh. Webwork: Meteor's web-based workflow management system. *Journal of Intelligent Information Systems*, 10(2):186–215, 1998.

[8] M. Nagarajan, K. Verma, A. P. Sheth, J. A. Miller, and J. Lathem. Semantic interoperability of web services - challenges and experiences. In *ICWS*, pages 373–382, 2006.

[9] N. Oldham, K. Verma, A. P. Sheth, and F. Hakimpour. Semantic ws-agreement partner selection. In *WWW*, pages 697–706, 2006.

[10] A. Sheth. Semantic web process lifecycle: Role of semantics in annotation, discovery, composition and orchestration, 2003.

[11] A. P. Sheth, K. Gomadam, and J. Lathem. Sa-rest: Semantically interoperable and easier-to-use services and mashups. *IEEE Internet Computing*, 11(6):91–94, 2007.

[12] K. Sivashanmugam, K. Verma, and A. P. Sheth. Discovery of web services in a federated registry environment. In *ICWS*, pages 270–278, 2004.

[13] K. Sivashanmugam, K. Verma, A. P. Sheth, and J. A. Miller. Adding semantics to web services standards. In *ICWS*, pages 395–401, 2003.

[14] K. Verma, P. Doshi, K. Gomadam, J. A. Miller, and A. P. Sheth. Optimal adaptation in web processes with coordination constraints. In *ICWS*, pages 257–264, 2006.

[15] K. Verma and A. P. Sheth. Semantically annotating a web service. *IEEE Internet Computing*, 11(2):83–85, 2007.